ΣΧΕΔΙΑΣΜΌΣ ΣΥΣΤΗΜΆΤΩΝ **VLSI**

$4^{η}$ Εργαστηριακή Άσκηση

Γιώργος Ντάκος 1059569

26 Μαΐου 2021

Περίληψη

Στο παρακάτω κείμενο παραδίδεται η αναφορά της $4^{ης}$ εργαστηριακής άσκησης του μαθήματος Σχεδιασμός Συστημάτων VLSI . Όλοι οι κώδικες έχουν συγγραφεί σε γλώσσα **VHDL** μέσω του **Notepad++** και έχουν μεταγλωτειστεί και επιβεβαιωθεί οι λειτουργίες τους μέσω του εργαλείου **ModelSim**.Τέλος η σύνθεση όλων των κυκλωμάτων πραγματοποιήθηκαν από το εργαλείο **VIVADO** της **Xilnix**.

Περιεχόμενα

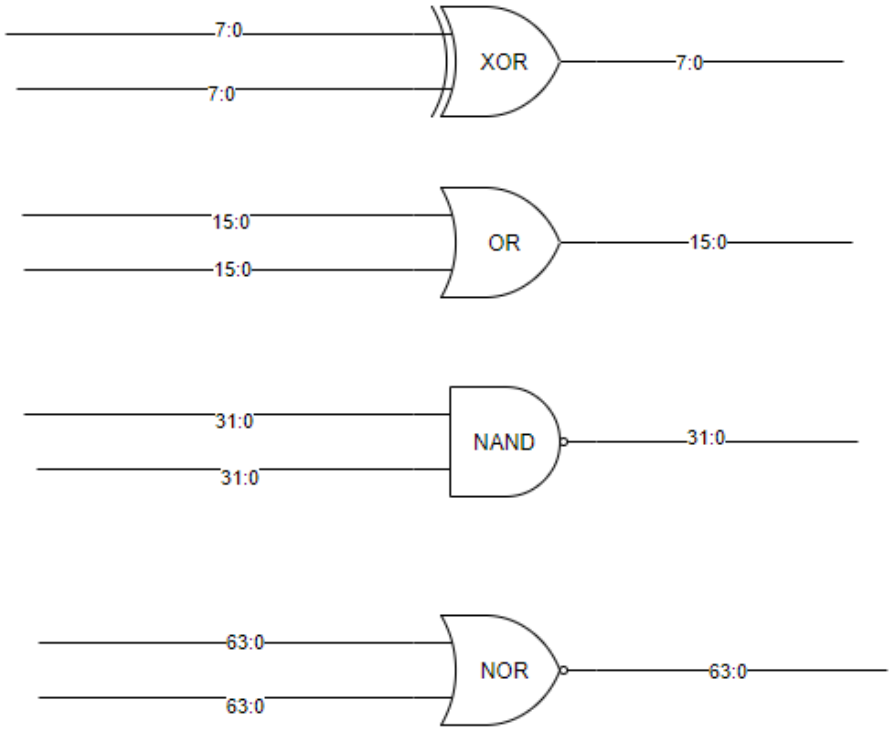
Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Ενότητα Α

Α.1,Α.2

Στην συγκεκριμένη ενότητα θα φτιάξουμε 4 λογικές πύλες δύο εισόδων και μιας εξόδου, (με διαφορετικά πλάτοι για την κάθε πύλη) οι οποίες φαίνονται στο παρακάτω σχηματικό και στον πίνακα 1 βρίσκονται οι πίνακες αληθείας τους

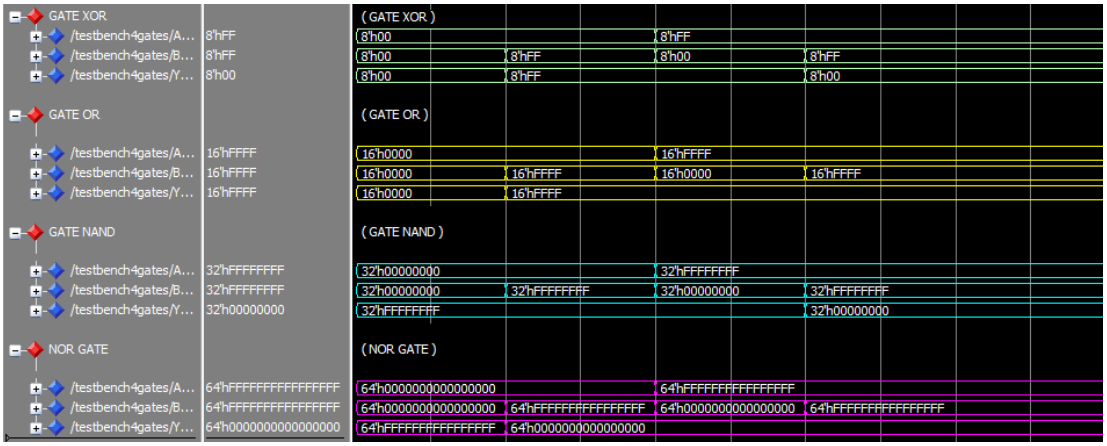


Σχήμα 1: Σχηματικό των λογικών πυλών

Στην συνέχεια ακολουθούν τα αποτελέσματα από το simulation μέσω ενός testbench για τις 4 λογικές πυλές και τα αποτελέσματα σύνθεσης για την κάθε μια.

GATE : XOR	IN_A	IN_B	OUT
	0	0	0
	0	1	1
	1	0	1
	1	1	0
GATE : OR	**IN_A**	**IN_B**	**OUT**
	0	0	0
	0	1	1
	1	0	1
	1	1	1
GATE : NAND	**IN_A**	**IN_B**	**OUT**
	0	0	1
	0	1	1
	1	0	1
	1	1	0
GATE : NOR	**IN_A**	**IN_B**	**OUT**
	0	0	1
	0	1	0
	1	0	0
	1	1	0

Table 1: Πίνακες αληθείας



Σχήμα 2: Αποτελέσματα simulation

POWER											
Name of circuit	Dynamic	Signals	Logic	I/O	Static	Junction Temperature	Power supplied to off-chip devices	Thermal Margin	Effective JA	Confidence level	Total On-Chip Power
xor8	3.553 W(98%)	0.097 W(3%)	0.02 W(1%)	3.437 W(96%)	0.067 W(2%)	47.4 C	0 W	52.6 C(8.5 W)	6.2 C/W	Low	3.62 W
or16	3.921 W(98%)	0.104 W(3%)	0.031 W(1%)	3.786 W(96%)	0.068 W(2%)	49.7 C	>>	50.3 C(8.1 W)	>>	>>	3.99 W
nand32	7.843 W(99%)	0.209 W(3%)	0.062 W(1%)	7.572 W(96%)	0.092 W(1%)	74.1 C	>>	25.9 C(4.2 W)	>>	>>	7.935 W
nor64	15.686 W(98%)	0.417 W(3%)	0.124 W(1%)	15.144 W(96%)	0.262 W(2%)	101.2 C	>>–	1.2 C (0.2 W)	4.8 C/W	>>	15.948 W

Table 2: PowerSupply

Utiliazation			
Name of circuit	**LUT**	**IO**	**FF**
xor8	8 of 8000 (0.1%)	24 of 112(21.43%)	0
or16	16 of 8000(0.2%)	48 of 112(42.86%)	>>
nand32	32 of 8000(0.4%)	96 of 112(85.71%)	>>
nor64	64 of 10400(0.62%)	192 of 210(91.43%)	>>

Table 3: Utiliazation

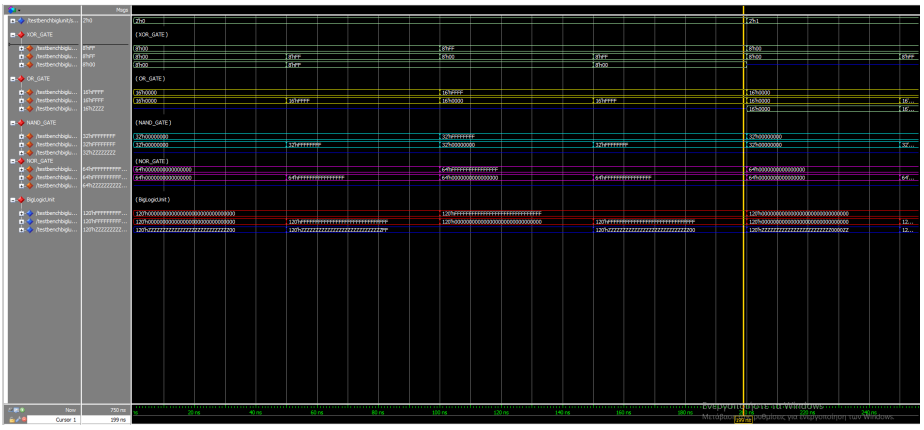
Timing				
	SETUP	**HOLD**	**TOTAL**	
	Total Delay	**Total Delay**	**Total Delay**	**Frequency**
xor8	4,574 ns	1,931 ns	6,505 ns	153,7 Mhz
or16	>>	>>	>>	>>
nand32	>>	>>	>>	>>
nor64	4,684 ns	2,04 ns	6,724 ns	148,72 MHz

Table 4: Data for Timing

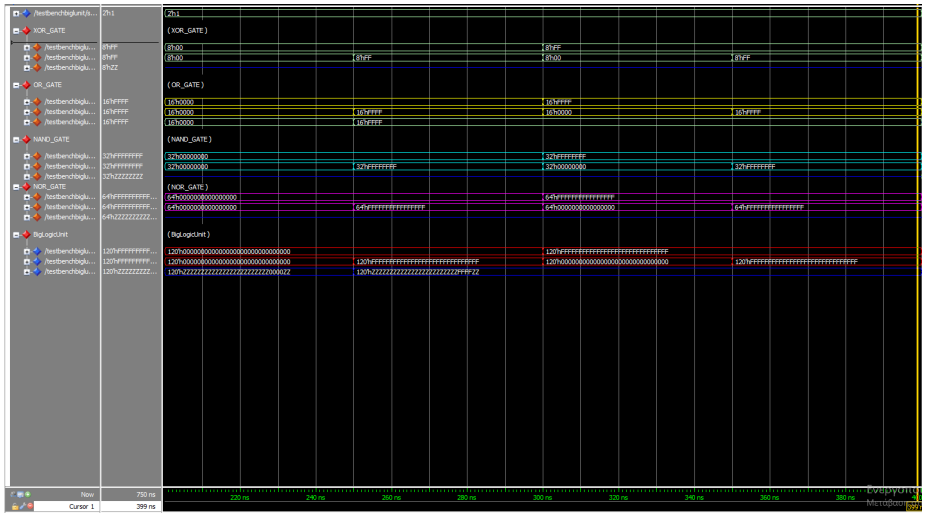
Ενότητα Β

Β.1,Β.2

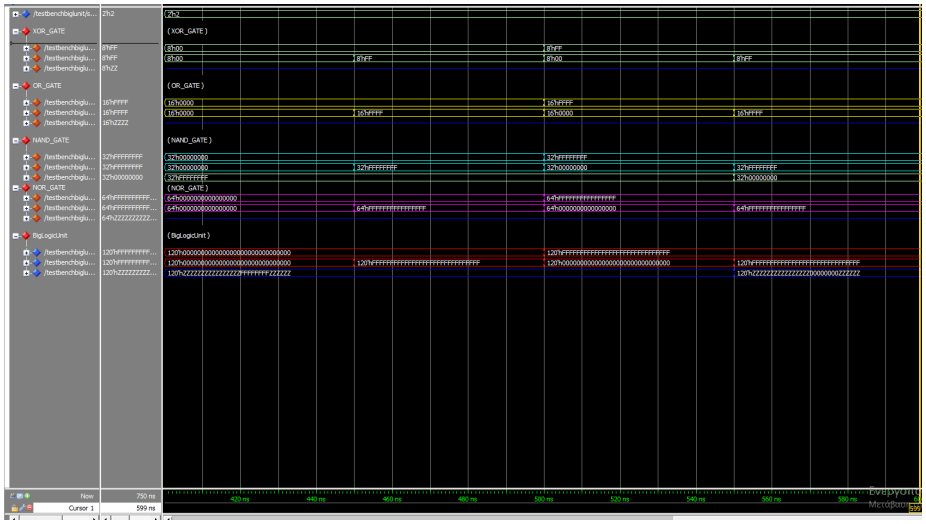
Στην συγκεκριμένη ενότητα θα φτιάξουμε μια μεγαλύτερη ενότητα η οποία θα περιέχει τις 4 λογικές πύλες που περιγράψαμε παραπάνω οπότε η μεγαλύτερη λογική μας πύλη θα έχει 2 εισόδους των 120 bits και 1 έξοδο των 120 bits . Επιπλέον θα έχουμε και ένα σήμα εισόδου των 2 bits για να επιλέγουμε ποιά πύλη θα εκτελείται κάθε φορά. Τώρα θα φτιάξουμε επιπλέον και έναν πολυπλέκτη 4 σε 1 ο οποίος θα περιέχεται μέσα στην μεγαλύτερη λογική μονάδα όπου οι 4 εισόδοι του θα οδηγούνται απο τις εξόδους των 4 λογικών πυλών και το σήμα επιλογής του θα οδηγείται απο το σήμα επιλογής της μεγαλύτερης λογικής μονάδας.Τέλος η έξοδος του πολυπλέκτη θα προωθείται στην έξοδο της μεγαλύτερης λογικής μονάδας. Στην συνέχεια ακολουθούν εικόνες από το σιμυλατιον του κυκλώματος το οποίο πραγματοποιήθηκε μέσω testbench και τα αποτελέσματα της σύνθεσης.



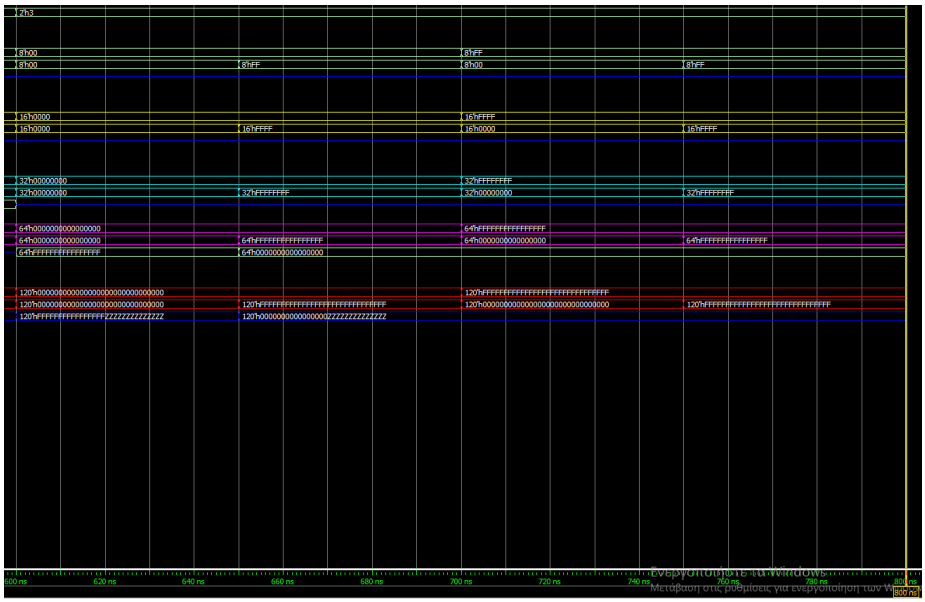
Σχήμα 3: Επιλογή πύλης XOR και τα αποτέλεσματα του simulation



Σχήμα 4: Επιλογή πύλης 0R και τα αποτελέσματα του simulation



Σχήμα 5: Επιλογή πύλης NAND και τα αποτελέσματα του simulation



Σχήμα 6: Επιλογή πύλης NOR και τα αποτελέσματα του simulation

POWER

Name of circuit	Dynamic	Signals	Logic	I/O	Static	Junction Temperature	Power supplied to off-chip devices	Thermal Margin	Effective JA	Confidence level	Total On-Chip Power
mux	3.599 W(98%)	0.095 W(3%)	0.03 W(1%)	3.474 W(96%)	0.067 W(2%)	47.7 C	0 W	52.3 C(8.4 W)	6.2 C/W	Low	3.666 W
bigLunit	20.736 W(99%)	0.590 W(3%)	0.577 W(3%)	19.570 W(94%)	0.223 W(1%)	76.1 C	0 W	23.9 C(9.6 W)	2.4 C/W	Low	20.96 W

Table 5: PowerSupply

Utiliazation			
Name of circuit	**LUT**	**IO**	**FF**
mux	8 of 8000 (0.1%)	42 of 112(37.5%)	0
bigLunit	600 of 48000(1.25%)	362 of 400(90.5%)	>>

Table 6: Utiliazation

Timing				
	SETUP	**HOLD**	**TOTAL**	
	Total Delay	**Total Delay**	**Total Delay**	**Frequency**
mux	4,574 ns	1,931 ns	6,505 ns	153,7 Mhz
bigLunit	5,177 ns	2,298 ns	7,475 ns	133,77 Mhz

Table 7: Data for Timing

Κώδικες σε VHDL

Παρακάτω παρατίθονται οι κώδικες της εργασίας(στο τέλος ακολουθούν τα σχηματικά σύμφωνα με την σύνθεση του κάθε κώδικα):

---

xor8

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY xor8 IS
GENERIC(N: NATURAL :=8);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END xor8;

 ARCHITECTURE structural OF xor8 IS
 BEGIN

 XORLoop: For i IN 0 TO N-1 Generate

 Y(i) <= A(i) XOR B(i);

 END Generate XORLoop;
 END structural;
```

---

or16

```
LIBRARY IEEE;
```

```
USE IEEE.std_logic_1164.all;

ENTITY or16 IS
GENERIC(N: NATURAL :=16);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END or16;

 ARCHITECTURE structural OF or16 IS
 BEGIN

 ORLoop: FOR i IN 0 TO N-1 GENERATE

 Y(i) <= A(i) OR B(i);

 END GENERATE ORLoop;
 END structural;
```

---

nand32

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY nand32 IS
GENERIC(N: NATURAL :=32);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END nand32;

 ARCHITECTURE structural OF nand32 IS
 BEGIN

 NANDLoop: For i IN 0 TO N-1 Generate

 Y(i) <= A(i) NAND B(i);
```

```
 END Generate NANDLoop;
 END structural;
```

---

nor64

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY nor64 IS
GENERIC(N: NATURAL :=64);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END nor64;

 ARCHITECTURE structural OF nor64 IS
 BEGIN

 NORLoop: FOR i IN 0 TO N-1 GENERATE

 Y(i) <= A(i) NOR B(i);

 END GENERATE NORLoop;
 END structural;
```

---

testbench4GATES

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY testbench4GATES IS
END testbench4GATES;

ARCHITECTURE testbench OF testbench4GATES IS
    COMPONENT xor8
    GENERIC(
        N: NATURAL :=8);
        PORT(
         A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
         B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
```

```
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

COMPONENT or16
GENERIC(
N: NATURAL :=16);
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

COMPONENT nand32
GENERIC(
N: NATURAL :=32);
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

COMPONENT nor64
GENERIC(
N: NATURAL :=64);
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

CONSTANT K : NATURAL := 8 ;
CONSTANT L : NATURAL := 16 ;
CONSTANT M : NATURAL := 32 ;
CONSTANT N : NATURAL := 64 ;

signal ATxor : STD_LOGIC_VECTOR(K-1 DOWNTO 0);
signal BTxor : STD_LOGIC_VECTOR(K-1 DOWNTO 0);
signal ATor : STD_LOGIC_VECTOR(L-1 DOWNTO 0);
signal BTor : STD_LOGIC_VECTOR(L-1 DOWNTO 0);
signal ATnand : STD_LOGIC_VECTOR(M-1 DOWNTO 0);
signal BTnand : STD_LOGIC_VECTOR(M-1 DOWNTO 0);
signal ATnor : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
```

```
        signal BTnor : STD_LOGIC_VECTOR(N-1 DOWNTO 0);

        signal YTxor : STD_LOGIC_VECTOR(K-1 DOWNTO 0);
        signal YTor : STD_LOGIC_VECTOR(L-1 DOWNTO 0);
        signal YTnand : STD_LOGIC_VECTOR(M-1 DOWNTO 0);
        signal YTnor : STD_LOGIC_VECTOR(N-1 DOWNTO 0);

BEGIN


        NOR_GATE: nor64 GENERIC MAP(N) PORT MAP(ATnor,BTnor,YTnor);
        NAND_GATE: nand32 GENERIC MAP(M) PORT MAP(ATnand,BTnand,YTnand);
        OR_GATE: or16 GENERIC MAP(L) PORT MAP (ATor,BTor,YTor);
        XOR_GATE: xor8 GENERIC MAP(K) PORT MAP(ATxor,BTxor,YTxor);

        stim_proc: process
        BEGIN

        ATxor<=X"00";
        BTxor<=X"00";

        ATor<=X"0000";
        BTor<=X"0000";

        ATnand<=X"00000000";
        BTnand<=X"00000000";

        ATnor<=X"0000000000000000";
        BTnor<=X"0000000000000000";

                WAIT FOR 20 ns;

        ATxor<=X"00";
        BTxor<=X"FF";

        ATor<=X"0000";
        BTor<=X"FFFF";

        ATnand<=X"00000000";
        BTnand<=X"FFFFFFFF";

        ATnor<=X"0000000000000000";
        BTnor<=X"FFFFFFFFFFFFFFFF";

                WAIT FOR 20 ns;
```

```
        ATxor<=X"FF";
        BTxor<=X"00";

        ATor<=X"FFFF";
        BTor<=X"0000";

        ATnand<=X"FFFFFFFF";
        BTnand<=X"00000000";

        ATnor<=X"FFFFFFFFFFFFFFFF";
        BTnor<=X"0000000000000000";

                WAIT FOR 20 NS;

        ATxor<=X"FF";
        BTxor<=X"FF";

        ATor<=X"FFFF";
        BTor<=X"FFFF";

        ATnand<=X"FFFFFFFF";
        BTnand<=X"FFFFFFFF";

        ATnor<=X"FFFFFFFFFFFFFFFF";
        BTnor<=X"FFFFFFFFFFFFFFFF";

                WAIT;

        END PROCESS;

END;
```

---

mux

```
--Dhlwsh bibliothhkhs kai twn paketwn poy tha xrhsimopoihsoyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY mux IS
GENERIC(N: NATURAL :=8);
PORT (
-- Shma eisodoy poy lambanei mia akoloythia timwn.
        a:IN std_logic_vector(N-1 downto 0);
```

```
-- Shma eisodoy poy lambanei mia akoloythia timwn.
        sel:IN std_logic_vector(1 downto 0);
-- Shma eisodoy poy lambanei mia akoloythia timwn.
        b :IN std_logic_vector(N-1 downto 0);
-- Shma eijodoy poy lambanei mia akoloythia timwn.

        c :IN STD_LOGIC_VECTOR(N-1 downto 0);

        d :IN STD_LOGIC_VECTOR(N-1 downto 0);

        x :OUT std_logic_vector(N-1 downto 0));
END mux;

--Dhlwsh arxitektonikhs
ARCHITECTURE behavioral OF mux IS
BEGIN
  --Ylopoihsh toy pinaka alhtheias
  --dld analoga me thn timh toy sel
  --pairnei kai antistoixh timh toy pinaka to shma ejodoy
        WITH (sel) SELECT
                x <= a WHEN "00",
                         b WHEN "01",
                         c WHEN "10",
                         d WHEN "11",
                    (OTHERS=>'Z') WHEN others;

END behavioral;
```

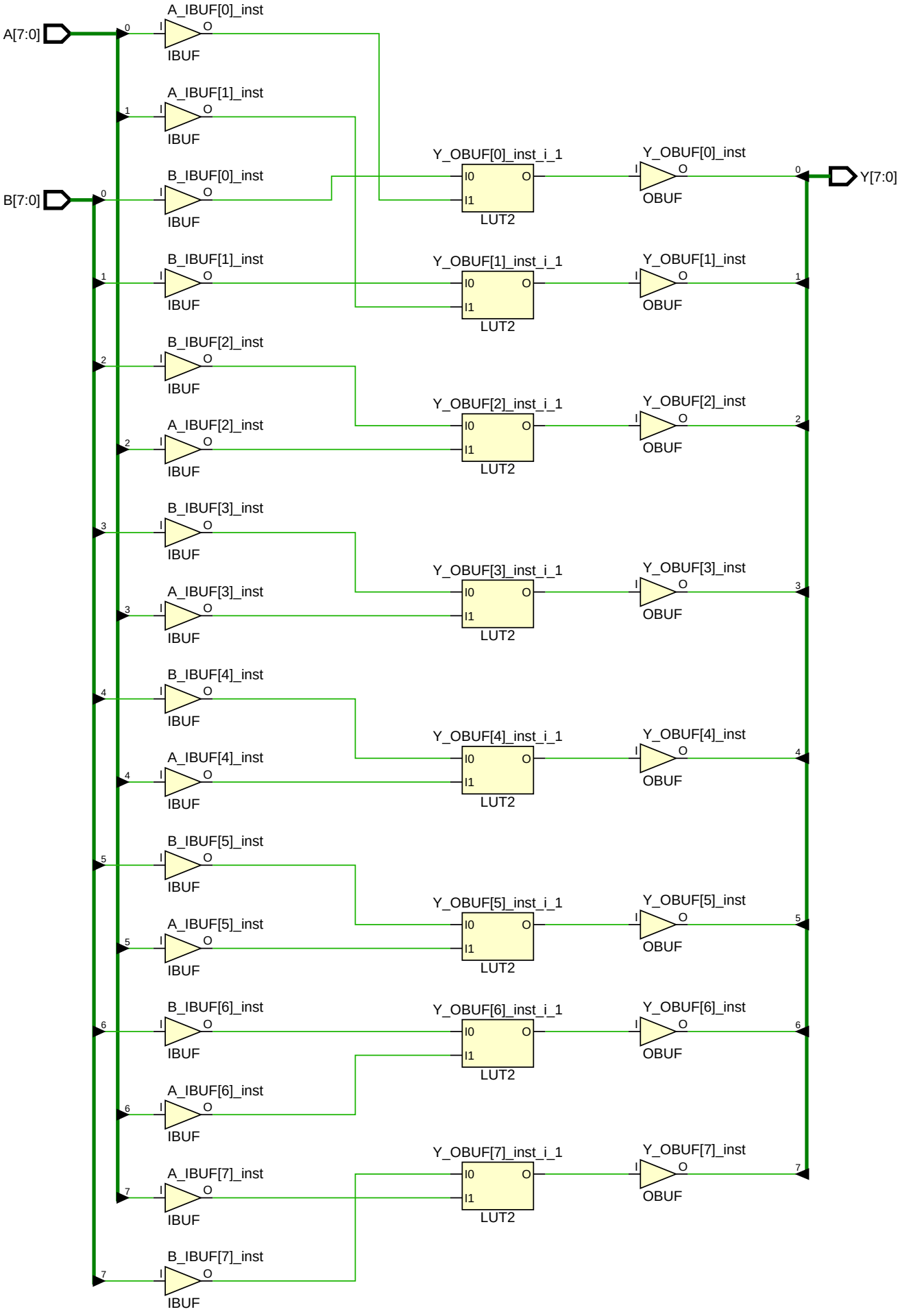
---

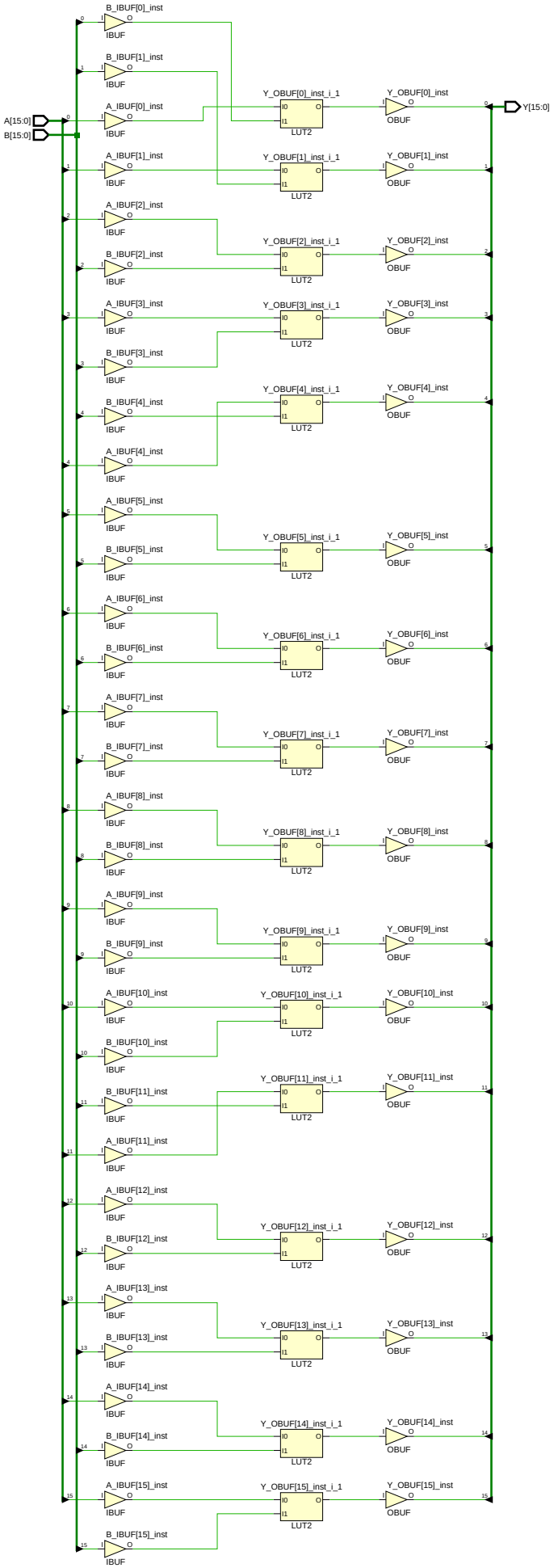
bigLunit

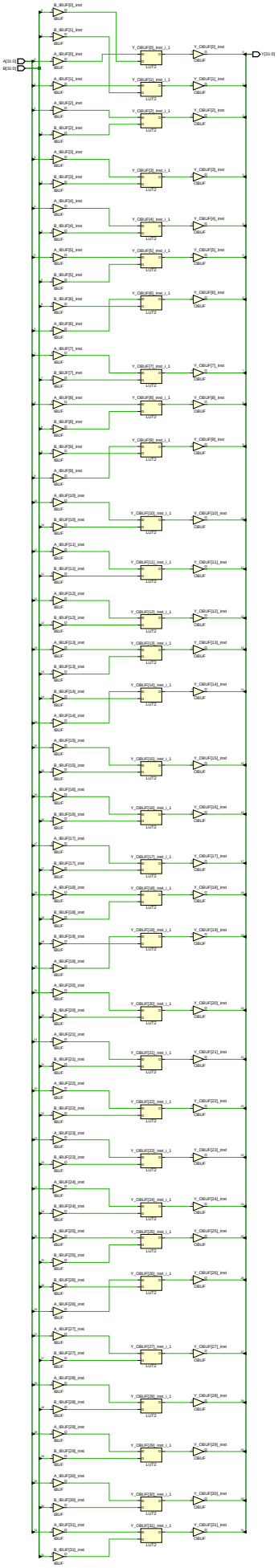
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY bigLunit IS
GENERIC(N: NATURAL :=120);
PORT(

 inp1 : IN std_logic_vector(N-1 DOWNTO 0);

 inp2 : IN std_logic_vector(N-1 DOWNTO 0);

 sel  : IN STD_LOGIC_VECTOR(1 DOWNTO 0);

 output : OUT std_logic_vector(N-1 DOWNTO 0));
 END bigLunit;
```

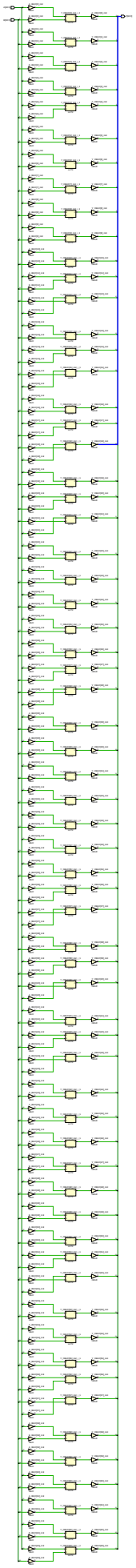
```
ARCHITECTURE structural OF bigLunit IS

COMPONENT xor8 IS
        GENERIC(N: NATURAL := 8);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT or16 IS
        GENERIC(N: NATURAL := 16);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT nand32 IS
        GENERIC(N: NATURAL := 32);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT nor64 IS
        GENERIC(N: NATURAL := 64);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT mux IS
        GENERIC(N: NATURAL := 8);
        PORT(     a : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  b : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  c : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  d : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
                  x : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

                  SIGNAL outXor : STD_LOGIC_VECTOR(N-113 DOWNTO 0);
                  SIGNAL outOr : STD_LOGIC_VECTOR(N-97 DOWNTO 8);
                  SIGNAL outNand : STD_LOGIC_VECTOR(N-65 DOWNTO 24);
                  SIGNAL outNOr : STD_LOGIC_VECTOR(N-1 DOWNTO 56);
```

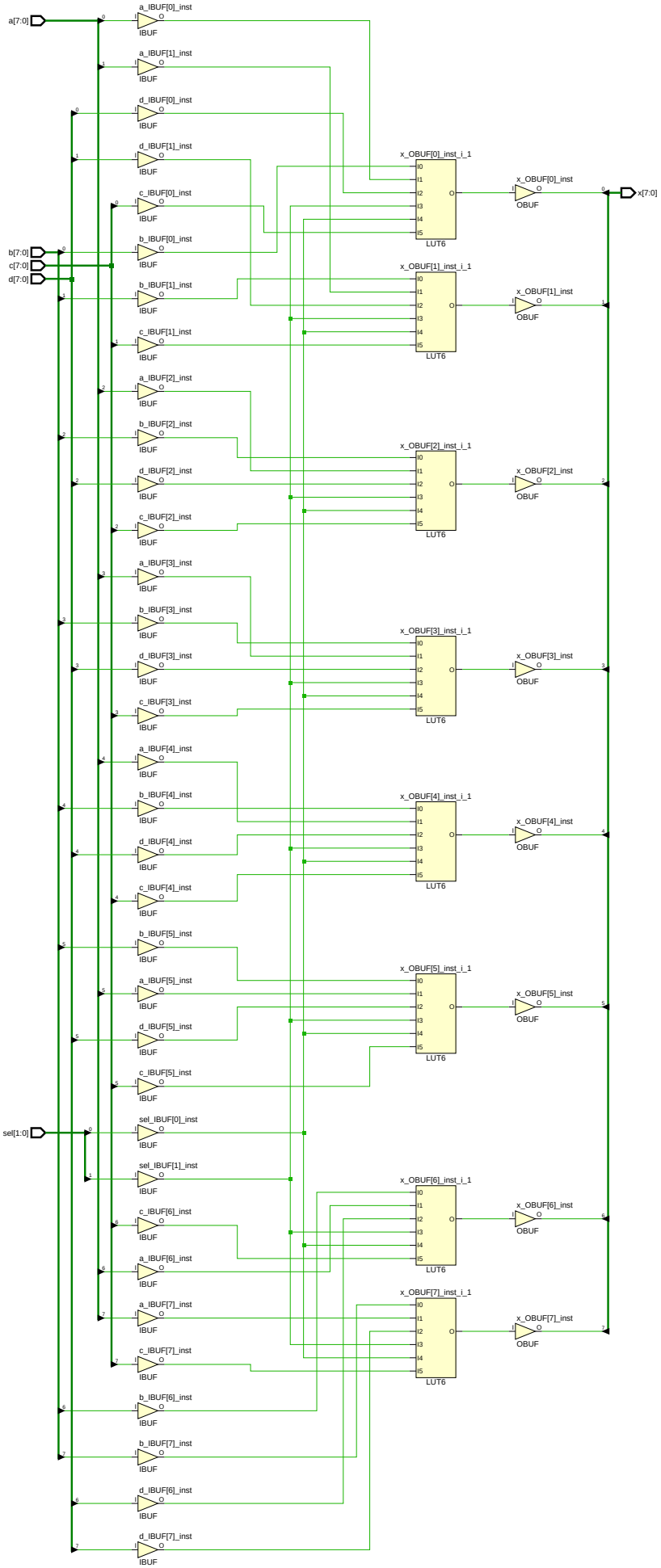
```
                SIGNAl in0_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                SIGNAL in1_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                SIGNAL in2_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                SIGNAL in3_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);

BEGIN
 XOR_GATE:  xor8   GENERIC MAP(N-112)  PORT MAP(inp1(N-113 DOWNTO 0),inp2(N-113 DOWNTO 0),ou
 OR_GATE:   or16   GENERIC MAP(N-104)  PORT MAP(inp1(N-97 DOWNTO 8),inp2(N-97 DOWNTO 8),out(
 NAND_GATE: nand32 GENERIC MAP(N-88)   PORT MAP(inp1(N-65 DOWNTO 24),inp2(N-65 DOWNTO 24),ou
 NOR_GATE:  nor64  GENERIC MAP(N-56)   PORT MAP(inp1(N-1 DOWNTO 56),inp2(N-1 DOWNTO 56 ),out

        in0_mux <= (N-1 DOWNTO 8 =>'Z') & outXor;
        in1_mux <= (N-1 DOWNTO 24 =>'Z') & outOr & (7 DOWNTO 0 => 'Z');
        in2_mux <= (N-1 DOWNTO 56 =>'Z') & outNand & (23 DOWNTO 0 =>'Z');
        in3_mux <= outNOr & (55 DOWNTO 0 =>'Z');


 MUX_GATE: mux     GENERIC MAP(N)      PORT MAP(in0_mux,in1_mux,in2_mux,in3_mux,sel,output);
END structural;
```


A[7:0]
B[7:0]
Y[7:0]
A_IBUF[0]_inst
A_IBUF[1]_inst
A_IBUF[2]_inst
A_IBUF[3]_inst
A_IBUF[4]_inst
A_IBUF[5]_inst
A_IBUF[6]_inst
A_IBUF[7]_inst
B_IBUF[0]_inst
B_IBUF[1]_inst
B_IBUF[2]_inst
B_IBUF[3]_inst
B_IBUF[4]_inst
B_IBUF[5]_inst
B_IBUF[6]_inst
B_IBUF[7]_inst
IBUF
Y_OBUF[0]_inst_i_1
Y_OBUF[1]_inst_i_1
Y_OBUF[2]_inst_i_1
Y_OBUF[3]_inst_i_1
Y_OBUF[4]_inst_i_1
Y_OBUF[5]_inst_i_1
Y_OBUF[6]_inst_i_1
Y_OBUF[7]_inst_i_1
LUT2
Y_OBUF[0]_inst
Y_OBUF[1]_inst
Y_OBUF[2]_inst
Y_OBUF[3]_inst
Y_OBUF[4]_inst
Y_OBUF[5]_inst
Y_OBUF[6]_inst
Y_OBUF[7]_inst
OBUF


A[15:0]
B[15:0]
Y[15:0]
B_IBUF[0]_inst
A_IBUF[0]_inst
Y_OBUF[0]_inst_i_1
Y_OBUF[0]_inst
B_IBUF[1]_inst
A_IBUF[1]_inst
Y_OBUF[1]_inst_i_1
Y_OBUF[1]_inst
A_IBUF[2]_inst
B_IBUF[2]_inst
Y_OBUF[2]_inst_i_1
Y_OBUF[2]_inst
A_IBUF[3]_inst
B_IBUF[3]_inst
Y_OBUF[3]_inst_i_1
Y_OBUF[3]_inst
B_IBUF[4]_inst
A_IBUF[4]_inst
Y_OBUF[4]_inst_i_1
Y_OBUF[4]_inst
A_IBUF[5]_inst
B_IBUF[5]_inst
Y_OBUF[5]_inst_i_1
Y_OBUF[5]_inst
A_IBUF[6]_inst
B_IBUF[6]_inst
Y_OBUF[6]_inst_i_1
Y_OBUF[6]_inst
A_IBUF[7]_inst
B_IBUF[7]_inst
Y_OBUF[7]_inst_i_1
Y_OBUF[7]_inst
A_IBUF[8]_inst
B_IBUF[8]_inst
Y_OBUF[8]_inst_i_1
Y_OBUF[8]_inst
A_IBUF[9]_inst
B_IBUF[9]_inst
Y_OBUF[9]_inst_i_1
Y_OBUF[9]_inst
A_IBUF[10]_inst
B_IBUF[10]_inst
Y_OBUF[10]_inst_i_1
Y_OBUF[10]_inst
B_IBUF[11]_inst
A_IBUF[11]_inst
Y_OBUF[11]_inst_i_1
Y_OBUF[11]_inst
A_IBUF[12]_inst
B_IBUF[12]_inst
Y_OBUF[12]_inst_i_1
Y_OBUF[12]_inst
A_IBUF[13]_inst
B_IBUF[13]_inst
Y_OBUF[13]_inst_i_1
Y_OBUF[13]_inst
A_IBUF[14]_inst
B_IBUF[14]_inst
Y_OBUF[14]_inst_i_1
Y_OBUF[14]_inst
A_IBUF[15]_inst
B_IBUF[15]_inst
Y_OBUF[15]_inst_i_1
Y_OBUF[15]_inst
IBUF
OBUF
LUT2
I0
I1
O
I


A[31:0]
B[31:0]
Y[31:0]
IBUF
OBUF
LUT2




a[7:0]
b[7:0]
c[7:0]
d[7:0]
sel[1:0]
x[7:0]
a_IBUF[0]_inst
a_IBUF[1]_inst
d_IBUF[0]_inst
d_IBUF[1]_inst
c_IBUF[0]_inst
b_IBUF[0]_inst
b_IBUF[1]_inst
c_IBUF[1]_inst
a_IBUF[2]_inst
b_IBUF[2]_inst
d_IBUF[2]_inst
c_IBUF[2]_inst
a_IBUF[3]_inst
b_IBUF[3]_inst
d_IBUF[3]_inst
c_IBUF[3]_inst
a_IBUF[4]_inst
b_IBUF[4]_inst
d_IBUF[4]_inst
c_IBUF[4]_inst
b_IBUF[5]_inst
a_IBUF[5]_inst
d_IBUF[5]_inst
c_IBUF[5]_inst
sel_IBUF[0]_inst
sel_IBUF[1]_inst
c_IBUF[6]_inst
a_IBUF[6]_inst
a_IBUF[7]_inst
c_IBUF[7]_inst
b_IBUF[6]_inst
b_IBUF[7]_inst
d_IBUF[6]_inst
d_IBUF[7]_inst
IBUF
x_OBUF[0]_inst_i_1
x_OBUF[1]_inst_i_1
x_OBUF[2]_inst_i_1
x_OBUF[3]_inst_i_1
x_OBUF[4]_inst_i_1
x_OBUF[5]_inst_i_1
x_OBUF[6]_inst_i_1
x_OBUF[7]_inst_i_1
I0
I1
I2
I3
I4
I5
O
LUT6
x_OBUF[0]_inst
x_OBUF[1]_inst
x_OBUF[2]_inst
x_OBUF[3]_inst
x_OBUF[4]_inst
x_OBUF[5]_inst
x_OBUF[6]_inst
x_OBUF[7]_inst
OBUF